

UNITED STATES PATENT APPLICATION

FOR

A METHOD AND OPERATION FOR PERFORMING MULTIPLY-ADD
OPERATIONS ON PACKED DATA

INVENTORS:

Alexander D. Peleg,
Millind Mittal,
Larry M. Mennemeier,
Benny Eitan,

Carole Dulong,
Eiichi Kowashi, and
Wolf Witt

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026

(408) 720-8598

File No. 42390.P2445

"Express Mail" mailing label number TB80583090505

Date of Deposit August 31, 1995

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Angela M. Quinn
(Typed or printed name of person mailing paper or fee)

Angela M. Quinn
(Signature of person mailing paper or fee)

A METHOD AND APPARATUS FOR PERFORMING MULTIPLY-ADD OPERATIONS ON PACKED DATA

5 CROSS-REFERENCE TO RELATED APPLICATION

Serial. No. _____, titled "A Method and Apparatus for Performing Multiply-Subtract Operations on Packed Data," filed _____, by Alexander D. Peleg, Millind Mittal, Larry M. Mennemeier, Benny Eitan, Andrew F. Glew, Carole Dulong, Eiichi Kowashi, and Wolf Witt.

10

BACKGROUND OF THE INVENTION

1. FIELD OF INVENTION

In particular, the invention relates to the field of computer systems. More specifically, the invention relates to the area of packed data operations.

15

2. DESCRIPTION OF RELATED ART

In typical computer systems, processors are implemented to operate on values represented by a large number of bits (e.g., 64) using instructions that produce one result. For example, the execution of an add instruction will add together a first 64-bit value and a second 64-bit value and store the result as a third 64-bit value. However, multimedia applications (e.g., applications targeted at computer supported cooperation (CSC -- the integration of teleconferencing with mixed media data manipulation), 2D/3D graphics, image processing, video compression/decompression, recognition algorithms and audio manipulation) require

the manipulation of large amounts of data which may be represented in a small number of bits. For example, graphical data typically requires 8 or 16 bits and sound data typically requires 8 or 16 bits. Each of these multimedia application requires one or more algorithms, each requiring a number of operations. For
5 example, an algorithm may require an add, compare and shift operation.

To improve efficiency of multimedia applications (as well as other applications that have the same characteristics), prior art processors provide packed data formats. A packed data format is one in which the bits typically used to represent a single value are broken into a number of fixed sized data elements, each
10 of which represents a separate value. For example, a 64-bit register may be broken into two 32-bit elements, each of which represents a separate 32-bit value. In addition, these prior art processors provide instructions for separately manipulating each element in these packed data types in parallel. For example, a packed add instruction adds together corresponding data elements from a first packed data and a
15 second packed data. Thus, if a multimedia algorithm requires a loop containing five operations that must be performed on a large number of data elements, it is desirable to pack the data and perform these operations in parallel using packed data instructions. In this manner, these processors can more efficiently process multimedia applications.

20 However, if the loop of operations contains an operation that cannot be performed by the processor on packed data (i.e., the processor lacks the appropriate instruction), the data will have to be unpacked to perform the operation. For example, if the multimedia algorithm requires an add operation and the previously

described packed add instruction is not available, the programmer must unpack both the first packed data and the second packed data (i.e., separate the elements comprising both the first packed data and the second packed data), add the separated elements together individually, and then pack the results into a packed result for further packed processing. The processing time required to perform such packing and unpacking often negates the performance advantage for which packed data formats are provided. Therefore, it is desirable to incorporate in a computer system a set of packed data instructions that provide all the required operations for typical multimedia algorithms. However, due to the limited die area on today's general purpose microprocessors, the number of instructions which may be added is limited. Therefore, it is desirable to invent instructions that provide both versatility (i.e. instructions which may be used in a wide variety of multimedia algorithms) and the greatest performance advantage.

One prior art technique for providing operations for use in multimedia algorithms is to couple a separate digital signaling processor (DSP) to an existing general purpose processor (e.g., The Intel® 486 manufactured by Intel Corporation of Santa Clara, CA). The general purpose processor allocates jobs that can be performed using packed data (e.g., video processing) to the DSP.

One such prior art DSP includes a multiply accumulate instruction that adds to an accumulation value the results of multiplying together two values. (see Kawakami, Yuichi, et al., "A Single-Chip Digital Signal Processor for Voiceband Applications", IEEE International Solid-State Circuits Conference, 1980, pp. 40-41). An example of the multiply accumulate operation for this DSP is shown below in

Table 1, where the instruction is performed on the data values A₁ and B₁ accessed as Source1 and Source2, respectively.

Multiply-Accumulate Source1, Source2	
A ₁	Source1
B ₁	Source2
=	
A ₁ B ₁ +Accumulation Value	Result1

Table 1

5

One limitation of this prior art instruction is its limited efficiency -- i.e., it only operates on 2 values and an accumulation value. For example, to multiply and accumulate two sets of 2 values requires the following 2 instructions performed serially: 1) multiply accumulate the first value from the first set, the first value from the second set, and an accumulation value of zero to generate an intermediate accumulation value; 2) multiply accumulate the second value from the first set, the second value from the second set, and the intermediate accumulation value to generate the result.

Another prior art DSP includes a multiply accumulate instruction that operates on two sets of two values and an accumulation value (See "Digital Signal Processor with Parallel Multipliers", patent number 4,771,470 - referred to herein as the "Ando et al." reference). An example of the multiply accumulate instruction for this DSP is shown below in Table 2, where the instruction is performed on the data values A₁, A₂, B₁ and B₂ accessed as Source1-4, respectively.

20

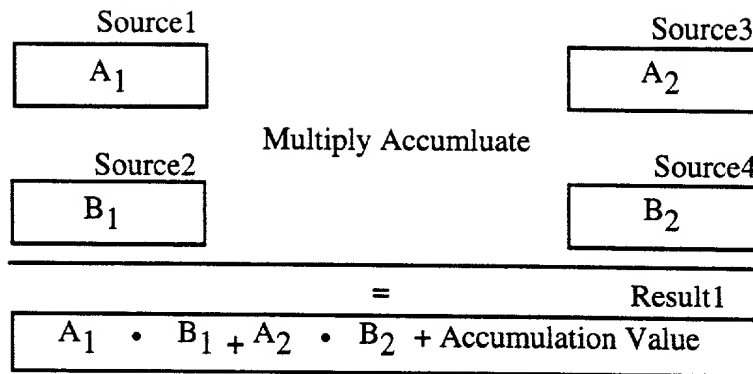


Table 2

Using this prior art technique, two sets of 2 values are multiplied and then added to
5 an accumulation value in one instruction.

This multiply accumulate instruction has limited versatility because it always
adds to the accumulation value. As a result, it is difficult to use the instruction for
operations other than multiply accumulate. For example, the multiplication of
complex numbers is commonly used in multimedia applications. The multiplication
10 of two complex number (e.g., $r_1 i_1$ and $r_2 i_2$) is performed according to the
following equation:

$$\text{Real Component} = r_1 \cdot r_2 - i_1 \cdot i_2$$

$$\text{Imaginary Component} = r_1 \cdot i_2 + r_2 \cdot i_1$$

This prior art DSP cannot perform the function of multiplying together two complex
15 numbers using one multiply accumulate instruction.

The limitations of this multiply accumulate instruction can be more clearly
seen when the result of such a calculation is needed in a subsequent multiplication
operation rather than an accumulation. For example, if the real component were

calculated using this prior art DSP, the accumulation value would need to be initialized to zero in order to correctly compute the result. Then the accumulation value would again need to be initialized to zero in order to calculate the imaginary component. To perform another complex multiplication on the resulting complex
5 number and a third complex number (e.g., r_3 , i_3), the resulting complex number must be rescaled and stored into the acceptable memory format and the accumulation value must again be initialized to zero. Then, the complex multiplication can be performed as described above. In each of these operations the ALU, which is devoted to the accumulation value, is superfluous hardware and extra
10 instructions are needed to re-initialize this accumulation value. These extra instructions would otherwise have been unnecessary.

A further limitation of this prior art technique is that the data must be accessed through expensive multi-ported memory. This is because the multipliers are connected directly with data memories. Therefore the amount of parallelism
15 which can be exploited is limited to a small number by the cost of the interconnection, and the fact that this interconnection is not decoupled from the instruction.

The Ando, et al. reference also describes that an alternative to this expensive interconnection is to introduce a delay for each subsequent pair of data to be
20 multiplied. This solution diminishes any performance advantages to those provided by the solution previously shown in Table 1.

Furthermore, the notion of multi-ported memory or of pipelined accesses to memory entails the use of multiple addresses. This explicit use of one address per

datum, clearly demonstrates that the critical notion of packed data is not employed in this prior art.

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235

SUMMARY OF THE INVENTION

A method and apparatus for including in a processor instructions for performing multiply-add operations on packed data is described. In one embodiment, a processor is coupled to a memory. The memory has stored therein a first packed data and a second packed data. The processor performs operations on data elements in the first packed data and the second packed data to generate a third packed data in response to receiving an instruction. At least two of the data elements in this third packed data storing the result of performing multiply-add operations on data elements in the first and second packed data.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is illustrated by way of example, and not limitation, in the figures. Like references indicate similar elements.

Figure 1 illustrates an exemplary computer system according to one
5 embodiment of the invention.

Figure 2 illustrates a register file of the processor according to one embodiment of the invention.

Figure 3 is a flow diagram illustrating the general steps used by the processor to manipulate data according to one embodiment of the invention.

10 Figure 4 illustrates packed data-types according to one embodiment of the invention.

Figure 5a illustrates in-register packed data representations according to one embodiment of the invention.

15 Figure 5b illustrates in-register packed data representations according to one embodiment of the invention.

Figure 5c illustrates in-register packed data representations according to one embodiment of the invention.

Figure 6a illustrates a control signal format for indicating the use of packed data according to one embodiment of the invention.

20 Figure 6b illustrates a second control signal format for indicating the use of packed data according to one embodiment of the invention.

Figure 7 is a flow diagram illustrating a method for performing multiply-add and multiply-subtract operations on packed data according to one embodiment of the invention.

Figure 8 illustrates a circuit for performing multiply-add and/or multiply-subtract operations on packed data according to one embodiment of the invention.

5

DETAILED DESCRIPTION

In the following description, numerous specific details are set forth to provide a thorough understanding of the invention. However, it is understood that the invention may be practiced without these specific details. In other instances, well-known circuits, structures and techniques have not been shown in detail in order not to obscure the invention.

DEFINITIONS

To provide a foundation for understanding the description of the embodiments of the invention, the following definitions are provided.

- 10 Bit X through Bit Y:
 defines a subfield of binary number. For example, bit six
 through bit zero of the byte 00111010₂ (shown in base
 two) represent the subfield 111010₂. The '2' following a
 binary number indicates base 2. Therefore, 1000₂ equals
15 8₁₀, while F₁₆ equals 15₁₀.
- R_X: is a register. A register is any device capable of storing
 and providing data. Further functionality of a register is
 described below. A register is not necessarily, included on
 the same die or in the same package as the processor..
- 20 SRC1, SRC2, and DEST:
 identify storage areas (e.g., memory addresses, registers,
 etc.)
- Source1-i and Result1-i:
 represent data.
- 25

OVERVIEW

This application describes a method and apparatus for including in a processor instructions for performing multiply-add and multiply-subtract operations on packed data. In one embodiment, two multiply-add operations are performed using a single multiply-add instruction as shown below in Table 3a and Table 3b -- Table 3a shows a simplified representation of the disclosed multiply-add instruction, while Table 3b shows a bit level example of the disclosed multiply-add instruction.

Multiply-Add Source1, Source2				
A1	A2	A3	A4	Source1
B1	B2	B3	B4	Source2
=				
A1B1+A2B2		A3B3+A4B4		Result1

Table 3a

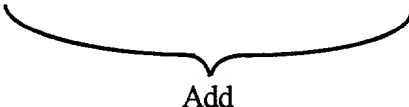
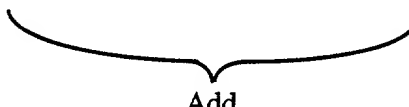
11111111 11111111	11111111 00000000	01110001 11000111	01110001 11000111
Multiply ³	Multiply ²	Multiply ¹	Multiply ⁰
00000000 00000000	00000000 00000001	10000000 00000000	00000100 00000000
↓	↓	↓	↓
32-Bit Intermediate Result 4	32-Bit Intermediate Result 3	32-Bit Intermediate Result 2	32-Bit Intermediate Result 1
			
11111111 11111111	11111111 00000000	11001000 11100011	10011100 00000000
1		0	

Table 3b

Thus, the described embodiment of the multiple-add instruction multiplies together corresponding 16-bit data elements of Source1 and Source2 generating four 32-bit intermediate results. These 32-bit intermediate results are summed by pairs producing two 32-bit results that are packed into their respective elements of a packed result. As further described later, alternative embodiment may vary the number of bits in the data elements, intermediate results, and results. In addition, alternative embodiment may vary the number of data elements used, the number of intermediate results generated, and the number of data elements in the resulting packed data. The multiply-subtract operation is the same as the multiply-add operation, except the adds are replaced with subtracts. The operation of an example multiply-subtract instruction is shown below in Table 4.

Multiply-Subtract Source1, Source2				
A1	A2	A3	A4	Source1
B1	B2	B3	B4	Source2
=				
A1B1-A2B2		A3B3-A4B4		Result1

Table 4

Of course, alternative embodiments may implement variations of these instructions. For example, alternative embodiments may include an instruction which performs at least one multiply-add operation or at least one multiply-subtract operation. As another example, alternative embodiments may include an instruction which performs at least one multiply-add operation in combination with at least one multiply-subtract operation. As another example, alternative embodiments may include an instruction which perform multiply-add operation(s) and/or multiply-subtract operation(s) in combination with some other operation.

COMPUTER SYSTEM

Figure 1 illustrates an exemplary computer system 100 according to one embodiment of the invention. Computer system 100 includes a bus 101, or other communications hardware and software, for communicating information, and a processor 109 coupled with bus 101 for processing information. Processor 109 represents a central processing unit of any type of architecture, including a CISC or RISC type architecture. Computer system 100 further includes a random access memory (RAM) or other dynamic storage device (referred to as main memory 104), coupled to bus 101 for storing information and instructions to be executed by processor 109. Main memory 104 also may be used for storing temporary variables

or other intermediate information during execution of instructions by processor 109. Computer system 100 also includes a read only memory (ROM) 106, and/or other static storage device, coupled to bus 101 for storing static information and instructions for processor 109. Data storage device 107 is coupled to bus 101 for
5 storing information and instructions.

Figure 1 also illustrates that processor 109 includes an execution unit 130, a register file 150, a cache 160, a decoder 165, and an internal bus 170. Of course, processor 109 contains additional circuitry which is not necessary to understanding the invention.

10 Execution unit 130 is used for executing instructions received by processor 109. In addition to recognizing instructions typically implemented in general purpose processors, execution unit 130 recognizes instructions in packed instruction set 140 for performing operations on packed data formats. Packed instruction set 140 includes instructions for supporting multiply-add and/or multiply-subtract
15 operations. In addition, packed instruction set 140 may also include instructions for supporting a pack operation, an unpack operation, a packed add operation, a packed subtract operation, a packed multiply operation, a packed shift operation, a packed compare operation, a population count operation, and a set of packed logical operations (including packed AND, packed ANDNOT, packed OR, and packed
20 XOR) as described in "A Set of Instructions for Operating on Packed Data," filed on _____, serial number _____.

Execution unit 130 is coupled to register file 150 by internal bus 170. Register file 150 represents a storage area on processor 109 for storing information, including

data. It is understood that one aspect of the invention is the described instruction set for operating on packed data. According to this aspect of the invention, the storage area used for storing the packed data is not critical. However, one embodiment of the register file 150 is later described with reference to Figure 2. Execution unit 130
5 is coupled to cache 160 and decoder 165. Cache 160 is used to cache data and/or control signals from, for example, main memory 104. Decoder 165 is used for decoding instructions received by processor 109 into control signals and/or microcode entry points. In response to these control signals and/or microcode entry points, execution unit 130 performs the appropriate operations. For example, if an
10 add instruction is received, decoder 165 causes execution unit 130 to perform the required addition; if a subtract instruction is received, decoder 165 causes execution unit 130 to perform the required subtraction; etc. Decoder 165 may be implemented using any number of different mechanisms (e.g., a look-up table, a hardware implementation, a PLA, etc.). Thus, while the execution of the various instructions
15 by the decoder and execution unit is represented by a series of if/then statements, it is understood that the execution of an instruction does not require a serial processing of these if/then statements. Rather, any mechanism for logically performing this if/then processing is considered to be within the scope of the invention.

Figure 1 additionally shows a data storage device 107, such as a magnetic disk
20 or optical disk, and its corresponding disk drive, can be coupled to computer system 100. Computer system 100 can also be coupled via bus 101 to a display device 121 for displaying information to a computer user. Display device 121 can include a frame buffer, specialized graphics rendering devices, a cathode ray tube (CRT),

and/or a flat panel display. An alphanumeric input device 122, including alphanumeric and other keys, is typically coupled to bus 101 for communicating information and command selections to processor 109. Another type of user input device is cursor control 123, such as a mouse, a trackball, a pen, a touch screen, or
5 cursor direction keys for communicating direction information and command selections to processor 109, and for controlling cursor movement on display device 121. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), which allows the device to specify positions in a plane. However, this invention should not be limited to input devices with only two
10 degrees of freedom.

Another device which may be coupled to bus 101 is a hard copy device 124 which may be used for printing instructions, data, or other information on a medium such as paper, film, or similar types of media. Additionally, computer system 100 can be coupled to a device for sound recording, and/or playback 125, such as an
15 audio digitizer coupled to a microphone for recording information. Further, the device may include a speaker which is coupled to a digital to analog (D/A) converter for playing back the digitized sounds.

Also, computer system 100 can be a terminal in a computer network (e.g., a LAN). Computer system 100 would then be a computer subsystem of a computer
20 network. Computer system 100 optionally includes video digitizing device 126. Video digitizing device 126 can be used to capture video images that can be transmitted to others on the computer network.

In one embodiment, the processor 109 additionally supports an instruction set which is compatible with the x86 instruction set used by existing processors (such as the Pentium® processor) manufactured by Intel Corporation of Santa Clara, California. Thus, in one embodiment, processor 109 supports all the operations
5 supported in the IA™ - Intel Architecture, as defined by Intel Corporation of Santa Clara, California (see Microprocessors, Intel Data Books volume 1 and volume 2, 1992 and 1993, available from Intel of Santa Clara, California). As a result, processor 109 can support existing x86 operations in addition to the operations of the invention. While the invention is described as being incorporated into an x86
10 based instruction set, alternative embodiments could incorporate the invention into other instruction sets. For example, the invention could be incorporated into a 64-bit processor using a new instruction set.

Figure 2 illustrates the register file of the processor according to one embodiment of the invention. The register file 150 is used for storing information,
15 including control/status information, integer data, floating point data, and packed data. In the embodiment shown in Figure 2, the register file 150 includes integer registers 201, registers 209, status registers 208, and instruction pointer register 211. Status registers 208 indicate the status of processor 109. Instruction pointer register 211 stores the address of the next instruction to be executed. Integer registers 201,
20 registers 209, status registers 208, and instruction pointer register 211 are all coupled to internal bus 170. Any additional registers would also be coupled to internal bus 170.

In one embodiment, the registers 209 are used for both packed data and floating point data. In one such embodiment, the processor 109, at any given time, must treat the registers 209 as being either stack referenced floating point registers or non-stack referenced packed data registers. In this embodiment, a mechanism is included
5 to allow the processor 109 to switch between operating on registers 209 as stack referenced floating point registers and non-stack referenced packed data registers. In another such embodiment, the processor 109 may simultaneously operate on registers 209 as non-stack referenced floating point and packed data registers. As another example, in another embodiment, these same registers may be used for
10 storing integer data.

Of course, alternative embodiments may be implemented to contain more or less sets of registers. For example, an alternative embodiment may include a separate set of floating point registers for storing floating point data. As another example, an alternative embodiment may including a first set of registers, each for
15 storing control/status information, and a second set of registers, each capable of storing integer, floating point, and packed data. As a matter of clarity, the registers of an embodiment should not be limited in meaning to a particular type of circuit. Rather, a register of an embodiment need only be capable of storing and providing data, and performing the functions described herein.

20 The various sets of registers (e.g., the integer registers 201, the registers 209) may be implemented to include different numbers of registers and/or to different size registers. For example, in one embodiment, the integer registers 201 are implemented to store thirty-two bits, while the registers 209 are implemented to

store eighty bits (all eighty bits are used for storing floating point data, while only sixty-four are used for packed data). In addition, registers 209 contains eight registers, R₀ 212a through R₇ 212h. R₁ 212a, R₂ 212b and R₃ 212c are examples of individual registers in registers 209. Thirty-two bits of a register in registers 209 can
5 be moved into an integer register in integer registers 201. Similarly, a value in an integer register can be moved into thirty-two bits of a register in registers 209. In another embodiment, the integer registers 201 each contain 64 bits, and 64 bits of data may be moved between the integer register 201 and the registers 209.

Figure 3 is a flow diagram illustrating the general steps are used by the
10 processor to manipulate data according to one embodiment of the invention. That is, Figure 3 illustrates the steps followed by processor 109 while performing an operation on packed data, performing an operation on unpacked data, or performing some other operation. For example, such operations include a load operation to load a register in register file 150 with data from cache 160, main memory 104, read only
15 memory (ROM) 106, or data storage device 107.

At step 301, the decoder 165 receives a control signal from either the cache 160 or bus 101. Decoder 165 decodes the control signal to determine the operations to be performed.

At step 302, Decoder 165 accesses the register file 150, or a location in
20 memory. Registers in the register file 150, or memory locations in the memory, are accessed depending on the register address specified in the control signal. For example, for an operation on packed data, the control signal can include SRC1, SRC2 and DEST register addresses. SRC1 is the address of the first source register.

SRC2 is the address of the second source register. In some cases, the SRC2 address is optional as not all operations require two source addresses. If the SRC2 address is not required for an operation, then only the SRC1 address is used. DEST is the address of the destination register where the result data is stored. In one
5 embodiment, SRC1 or SRC2 is also used as DEST. SRC1, SRC2 and DEST are described more fully in relation to Figure 6a and Figure 6b. The data stored in the corresponding registers is referred to as Source1, Source2, and Result respectively. Each of these data is sixty-four bits in length.

In another embodiment of the invention, any one, or all, of SRC1, SRC2 and
10 DEST, can define a memory location in the addressable memory space of processor 109. For example, SRC1 may identify a memory location in main memory 104, while SRC2 identifies a first register in integer registers 201 and DEST identifies a second register in registers 209. For simplicity of the description herein, the invention will be described in relation to accessing the register file 150. However,
15 these accesses could be made to memory instead.

At step 303, execution unit 130 is enabled to perform the operation on the accessed data. At step 304, the result is stored back into register file 150 according to requirements of the control signal.

DATA AND STORAGE FORMATS

20 Figure 4 illustrates packed data-types according to one embodiment of the invention. Three packed data formats are illustrated; packed byte 401, packed word 402, and packed doubleword 403. Packed byte, in one embodiment of the invention, is sixty-four bits long containing eight data elements. Each data element is one byte

long. Generally, a data element is an individual piece of data that is stored in a single register (or memory location) with other data elements of the same length. In one embodiment of the invention, the number of data elements stored in a register is sixty-four bits divided by the length in bits of a data element.

5 Packed word 402 is sixty-four bits long and contains four word 402 data elements. Each word 402 data element contains sixteen bits of information.

 Packed doubleword 403 is sixty-four bits long and contains two doubleword 403 data elements. Each doubleword 403 data element contains thirty-two bits of information.

10 Figure 5a through 5c illustrate the in-register packed data storage representation according to one embodiment of the invention. Unsigned packed byte in-register representation 510 illustrates the storage of an unsigned packed byte 401 in one of the registers R0 212a through R7 212h. Information for each byte data element is stored in bit seven through bit zero for byte zero, bit fifteen through bit eight for byte
15 one, bit twenty-three through bit sixteen for byte two, bit thirty-one through bit twenty-four for byte three, bit thirty-nine through bit thirty-two for byte four, bit forty-seven through bit forty for byte five, bit fifty-five through bit forty-eight for byte six and bit sixty-three through bit fifty-six for byte seven. Thus, all available
20 bits are used in the register. This storage arrangement increases the storage efficiency of the processor. As well, with eight data elements accessed, one operation can now be performed on eight data elements simultaneously. Signed packed byte in-register representation 511 illustrates the storage of a signed packed byte 401. Note that the eighth bit of every byte data element is the sign indicator.

Unsigned packed word in-register representation 512 illustrates how word three through word zero are stored in one register of registers 209. Bit fifteen through bit zero contain the data element information for word zero, bit thirty-one through bit sixteen contain the information for data element word one, bit forty-seven through bit thirty-two contain the information for data element word two and bit sixty-three through bit forty-eight contain the information for data element word three. Signed packed word in-register representation 513 is similar to the unsigned packed word in-register representation 512. Note that the sixteenth bit of each word data element is the sign indicator.

Unsigned packed doubleword in-register representation 514 shows how registers 209 store two doubleword data elements. Doubleword zero is stored in bit thirty-one through bit zero of the register. Doubleword one is stored in bit sixty-three through bit thirty-two of the register. Signed packed doubleword in-register representation 515 is similar to unsigned packed doubleword in-register representation 514. Note that the necessary sign bit is the thirty-second bit of the doubleword data element.

As mentioned previously, registers 209 may be used for both packed data and floating point data. In this embodiment of the invention, the individual programming processor 109 may be required to track whether an addressed register, R₀ 212a for example, is storing packed data or floating point data. In an alternative embodiment, processor 109 could track the type of data stored in individual registers of registers 209. This alternative embodiment could then generate errors if, for example, a packed addition operation were attempted on floating point data.

CONTROL SIGNAL FORMATS

The following describes one embodiment of the control signal formats used by processor 109 to manipulate packed data. In one embodiment of the invention, control signals are represented as thirty-two bits. Decoder 165 may receive the
5 control signal from bus 101. In another embodiment, decoder 165 can also receive such control signals from cache 160.

Figure 6a illustrates a control signal format for indicating the use of packed data according to one embodiment of the invention. Operation field OP 601, bit thirty-one through bit twenty-six, provides information about the operation to be
10 performed by processor 109; for example, packed addition, packed subtraction, etc.. SRC1 602, bit twenty-five through twenty, provides the source register address of a register in registers 209. This source register contains the first packed data, Source1, to be used in the execution of the control signal. Similarly, SRC2 603, bit nineteen through bit fourteen, contains the address of a register in registers 209. This second
15 source register contains the packed data, Source2, to be used during execution of the operation. DEST 605, bit five through bit zero, contains the address of a register in registers 209. This destination register will store the result packed data, Result, of the packed data operation.

Control bits SZ 610, bit twelve and bit thirteen, indicates the length of the data
20 elements in the first and second packed data source registers. If SZ 610 equals 01₂, then the packed data is formatted as packed byte 401. If SZ 610 equals 10₂, then the packed data is formatted as packed word 402. SZ 610 equaling 00₂ or 11₂ is

reserved, however, in another embodiment, one of these values could be used to indicate packed doubleword 403.

Control bit T 611, bit eleven, indicates whether the operation is to be carried out with saturate mode. If T 611 equals one, then a saturating operation is performed. If
5 T 611 equals zero, then a non-saturating operation is performed. Saturating operations will be described later.

Control bit S 612, bit ten, indicates the use of a signed operation. If S 612 equals one, then a signed operation is performed. If S 612 equals zero, then an unsigned operation is performed.

10 Figure 6b illustrates a second control signal format for indicating the use of packed data according to one embodiment of the invention. This format corresponds with the general integer opcode format described in the "Pentium Processor Family User's Manual," available from Intel Corporation, Literature Sales, P.O. Box 7641, Mt. prospect, IL, 60056-7641. Note that OP 601, SZ 610, T 611, and S 612 are all
15 combined into one large field. For some control signals, bits three through five are SRC1 602. In one embodiment, where there is a SRC1 602 address, then bits three through five also correspond to DEST 605. In an alternate embodiment, where there is a SRC2 603 address, then bits zero through two also correspond to DEST 605. For other control signals, like a packed shift immediate operation, bits three through five
20 represent an extension to the opcode field. In one embodiment, this extension allows a programmer to include an immediate value with the control signal, such as a shift count value. In one embodiment, the immediate value follows the control signal. This is described in more detail in the "Pentium Processor Family User's Manual,"

in appendix F, pages F-1 through F-3. Bits zero through two represent SRC2 603.

This general format allows register to register, memory to register, register by memory, register by register, register by immediate, register to memory addressing.

Also, in one embodiment, this general format can support integer register to register,

5 and register to integer register addressing.

DESCRIPTION OF SATURATE/UNSATURATE

As mentioned previously, T 611 indicates whether operations optionally saturate. Where the result of an operation, with saturate enabled, overflows or underflows the range of the data, the result will be clamped. Clamping means setting
10 the result to a maximum or minimum value should a result exceed the range's maximum or minimum value. In the case of underflow, saturation clamps the result to the lowest value in the range and in the case of overflow, to the highest value. The allowable range for each data format is shown in Table 5.

Data Format	Minimum Value	Maximum Value
Unsigned Byte	0	255
Signed Byte	-128	127
Unsigned Word	0	65535
Signed Word	-32768	32767
Unsigned Doubleword	0	2 ⁶⁴ -1
Signed Doubleword	-2 ⁶³	2 ⁶³ -1

Table 5

As mentioned above, T 611 indicates whether saturating operations are being performed. Therefore, using the unsigned byte data format, if an operation's result = 258 and saturation was enabled, then the result would be clamped to 255 before being stored into the operation's destination register. Similarly, if an operation's

result = -32999 and processor 109 used signed word data format with saturation enabled, then the result would be clamped to -32768 before being stored into the operation's destination register.

MULTIPLY-ADD/SUBTRACT OPERATION(S)

5 In one embodiment of the invention, the SRC1 register contains packed data (Source1), the SRC2 register contains packed data (Source2), and the DEST register will contain the result (Result) of performing the multiply-add or multiply-subtract instruction on Source1 and Source2. In the first step of the multiply-add and multiply-subtract instruction, Source1 will have each data element independently
10 multiplied by the respective data element of Source2 to generate a set of respective intermediate results. These intermediate results are summed by pairs to generate the Result for the multiply-add instruction. In contrast, these intermediate results are subtracted by pairs to generate the Result for the multiply-subtract instruction.

In one embodiment of the invention, the multiply-add and multiply-subtract
15 instructions operate on signed packed data and truncate the results to avoid any overflows. In addition, these instructions operate on packed word data and the Result is a packed double word. However, alternative embodiments could support these instructions for other packed data types.

Figure 7 is a flow diagram illustrating a method for performing multiply-add
20 and multiply-subtract operations on packed data according to one embodiment of the invention.

At step 701, decoder 165 decodes the control signal received by processor 109. Thus, decoder 165 decodes: the operation code for a multiply-add instruction or a multiply-subtract instruction.

At step 702, via internal bus 170, decoder 165 accesses registers 209 in register
5 file 150 given the SRC1 602 and SRC2 603 addresses. Registers 209 provide execution unit 130 with the packed data stored in the SRC1 602 register (Source1), and the packed data stored in SRC2 603 register (Source2). That is, registers 209 communicate the packed data to execution unit 130 via internal bus 170.

At step 703, decoder 165 enables execution unit 130 to perform the instruction.
10 If the instruction is a multiply-add instruction, flow passes to step 714. However, if the instruction is a multiply-subtract instruction, flow passes to step 715.

In step 714, the following is performed. Source1 bits fifteen through zero are multiplied by Source2 bits fifteen through zero generating a first 32-bit intermediate result (Intermediate Result 1). Source1 bits thirty-one through sixteen are multiplied
15 by Source2 bits thirty-one through sixteen generating a second 32-bit intermediate result (Intermediate Result 2). Source1 bits forty-seven through thirty-two are multiplied by Source2 bits forty-seven through thirty-two generating a third 32-bit intermediate result (Intermediate Result 3). Source1 bits sixty-three through forty-eight are multiplied by Source2 bits sixty-three through forty-eight generating a
20 fourth 32-bit intermediate result (Intermediate Result 4). Intermediate Result 1 is added to Intermediate Result 2 generating Result bits thirty-one through 0, and Intermediate Result 3 is added to Intermediate Result 4 generating Result bits sixty-three through thirty-two.

Step 715 is the same as step 714, with the exception that Intermediate Result 1 Intermediate Result 2 are subtracted to generate bits thirty-one through 0 of the Result, and Intermediate Result 3 and Intermediate Result 4 are subtracted to generate bits sixty-three through thirty-two of the Result.

- 5 Different embodiments may perform the multiplies and adds/subtracts serially, in parallel, or in some combination of serial and parallel operations.

At step 720, the Result is stored in the DEST register.

PACKED DATA MULTIPLY-ADD/SUBTRACT CIRCUITS

- 10 In one embodiment, the multiply-add and multiply-subtract instructions can execute on multiple data elements in the same number of clock cycles as a single multiply on unpacked data. To achieve execution in the same number of clock cycles, parallelism is used. That is, registers are simultaneously instructed to perform the multiply-add/subtract operations on the data elements. This is discussed in more detail below.

- 15 Figure 8 illustrates a circuit for performing multiply-add and/or multiply-subtract operations on packed data according to one embodiment of the invention. Operation control 800 processes the control signal for the multiply-add and multiply-subtract instructions. Operation control 800 outputs signals on Enable 880 to control Packed multiply-adder/subtractor 801.

- 20 Packed multiply-adder/subtractor 801 has the following inputs: Source1[63:0] 831, Source2[63:0] 833, and Enable 880. Packed multiply-adder/subtractor 801 includes four 16x16 multiplier circuits: 16x16 multiplier A 810, 16x16 multiplier B 811, 16x16 multiplier C 812 and 16x16 multiplier D 813. 16x16 multiplier A 810

has as inputs Source1[15:0] and Source2[15:0]. 16x16 multiplier B 811 has as inputs Source1[31:16] and Source2[31:16]. 16x16 multiplier C 812 has as inputs Source1[47:32] and Source2[47:32]. 16x16 multiplier D 813 has as inputs Source1[63:48] and Source2[63:48]. The 32-bit intermediate results generated by

5 16x16 multiplier A 810 and 16x16 multiplier B 811 are received by adder/subtractor 1350, while the 32-bit intermediate results generated by 16x16 multiplier C 812 and 16x16 multiplier D 813 are received by adder/subtractor 851.

Based on whether the current instruction is a multiply/add or multiply/subtract instruction, adder/subtractor 850 and adder/subtractor 851 add or subtract their

10 respective 32-bit inputs. The output of adder/subtractor 850 (i.e., Result bits 31 through zero of the Result) and the output of adder/subtractor 851 (i.e., bits 63 through 32 of the Result) are combined into the 64-bit Result and communicated to Result Register 871.

In one embodiment, each of adder/subtractor 851 and adder/subtractor 850 are

15 composed of four 8-bit adders/subtractors with the appropriate propagation delays. However, alternative embodiments could implement adder/subtractor 851 and adder/subtractor 850 in any number of ways (e.g., two 32-bit adders/subtractors).

To perform the equivalent of these multiply-add or multiply-subtract instructions in prior art processors which operate on unpacked data, four separate

20 64-bit multiply operations and two 64-bit add or subtract operations, as well as the necessary load and store operations, would be needed. This wastes data lines and circuitry that are used for the bits that are higher than bit sixteen for Source1 and Source 2, and higher than bit thirty two for the Result. As well, the entire 64-bit

result generated by the prior art processor may not be of use to the programmer.

Therefore, the programmer would have to truncate each result.

Performing the equivalent of this multiply-add instruction using the prior art DSP processor described with reference to Table 1 requires one instruction to zero the accumulation value and four multiply accumulate instructions. Performing the equivalent of this multiply-add instruction using the prior art DSP processor described with reference to Table 2 requires one instruction to zero the accumulation value and 2-accumulate instructions.

10 ADVANTAGES OF INCLUDING THE DESCRIBED MULTIPLY-ADD INSTRUCTION
 IN THE INSTRUCTION SET

As previously described, the prior art multiply accumulate instructions always add the results of their multiplications to an accumulation value. This accumulation value becomes a bottleneck for performing operations other than multiplying and accumulating (e.g., the accumulation value must be cleared each time a new set of operations is required which do not require the previous accumulation value). This accumulation value also becomes a bottleneck if operations, such as rounding, need to be performed before accumulation.

20 In contrast, the disclosed multiply-add and multiply-subtract instructions do not carry forward an accumulation value. As a result, these instructions are easier to use in a wider variety of algorithms. In addition, software pipelining can be used to achieve comparable throughput. To illustrate the versatility of the multiply-add instruction, several example multimedia algorithms are described below. Some of these multimedia algorithms use additional packed data instructions. The operation

of these additional packed data instructions are shown in relation to the described algorithms. For a further description of these packed data instructions, see "A Set of Instructions for Operating on Packed Data," filed on _____, serial number _____ . Of course, other packed data instructions could be used. In

5 addition, a number of steps requiring the use of general purpose processor instructions to manage data movement, looping, and conditional branching have been omitted in the following examples.

1) Multiplication of Complex Numbers

The disclosed multiply-add instruction can be used to multiply two complex
10 numbers in a single instruction as shown in Table 6a. As previously described, the multiplication of two complex number (e.g., $r_1 i_1$ and $r_2 i_2$) is performed according to the following equation:

$$\text{Real Component} = r_1 \cdot r_2 - i_1 \cdot i_2$$

$$\text{Imaginary Component} = r_1 \cdot i_2 + r_2 \cdot i_1$$

15 If this instruction is implemented to be completed every clock cycle, the invention can multiply two complex numbers every clock cycle.

Multiply-Add Source1, Source2				
r1	i2	r1	i1	Source1
r2	-i2	i2	r2	Source2
=				
Real Component: $r_1 r_2 - i_1 i_2$		Imaginary Component: $r_1 i_2 + r_2 i_1$		Result 1

Table 6a

20 As another example, Table 6b shows the instructions used to multiply together three complex numbers.

Multiply-Add Source1, Source2				
r1	i1	r1	i1	Source1
r2	-i2	i2	r2	Source2
=				
Real Component1: r1r2-i1i2		Imaginary Component1: r1i2+r2i1		Result1

5

Packed Shift Right Source1, Source2				
Real Component ₁		Imaginary Component ₁		Result1
16				
=				
	Real Component ₁		Imaginary Component ₁	Result2

Pack Result2, Result2				
	Real Component1		Imaginary Component1	Result2
	Real Component1		Imaginary Component1	Result2
=				
Real Component1	Imaginary Component1	Real Component1	Imaginary Component1	Result3

10

Multiply-Add Result3, Source3				
Real Component1: r1r2-i1i2	Imaginary Component1: r1i2+r2i1	Real Component1: r1r2-i1i2	Imaginary Component1: r1i2+r2i1	Result3
r3	-i3	i3	r3	Source3
=				
Real Component2		Imaginary Component2		Result4

Table 6b

2) Multiply Accumulation Operations

The disclosed multiply-add instructions can also be used to multiply and accumulate values. For example, two sets of four data elements (A_{1-4} and B_{1-4}) may be multiplied and accumulated as shown below in Table 7. In one embodiment, each of the instructions shown in Table 7 is implemented to complete each clock cycle.

Multiply-Add Source1, Source2				
0	0	A_1	A_2	Source1
0	0	B_1	B_2	Source2
=				
0	$A_1B_1+A_2B_2$			Result1

Multiply-Add Source3, Source4				
0	0	A_3	A_4	Source3
0	0	B_3	B_4	Source4
=				
0	$A_3A_4+B_3B_4$			Result2

Unpacked Add Result1, Result2		
0	$A_1B_1+A_2B_2$	Result1
0	$A_3A_4+B_3B_4$	Result2
=		
0	$A_1B_1+A_2B_2+A_3A_4+B_3B_4$	Result3

Table 7

If the number of data elements in each set exceeds 8 and is a multiple of 4, the multiplication and accumulation of these sets requires fewer instructions if performed as shown in table 8 below.

Multiply-Add Source1, Source2

A1	A2	A3	A4	Source1
B1	B2	B3	B4	Source2
=				
A1B1+A2B2		A3B3+A4B4		Result1

5

Multiply-Add Source3, Source4

A5	A6	A7	A8	Source3
B5	B6	B7	B8	Source4
=				
A5B5+A6B6		A7B7+A8B8		Result2

Packed Add Result1, Result2

A1B1+A2B2		A3B3+A4B4	Result1
A5B5+A6B6		A7B7+A8B8	Result2
=			
A1B1+A2B2+A5B5+A6B6		A3B3+A4B4+A7B7+A8B8	Result3

10

Unpack High Result3, Source5

A1B1+A2B2+A5B5+A6B6		A3B3+A4B4+A7B7+A8B8	Result3
0		0	Source5
		=	
0		A1B1+A2B2+A5B5+A6B6	Result4

Unpack Low Result3, Source5		
$A_1B_1+A_2B_2+A_5B_5+A_6B_6$	$A_3B_3+A_4B_4+A_7B_7+A_8B_8$	Result3
0	0	Source5
=		
0	$A_3B_3+A_4B_4+A_7B_7+A_8B_8$	Result5

Packed Add Result4, Result5		
0	$A_1B_1+A_2B_2+A_5B_5+A_6B_6$	Result4
0	$A_3B_3+A_4B_4+A_7B_7+A_8B_8$	Result5
=		
0	TOTAL	Result6

Table 8

As another example, Table 9 shows the separate multiplication and accumulation of sets A and B and sets C and D, where each of these sets includes 2 data elements.

Multiply-Add Source1, Source2				
A ₁	A ₂	C ₁	C ₂	Source1
B ₁	B ₂	D ₁	D ₂	Source2
=				
$A_1B_1+A_2B_2$		$C_1D_1+C_2D_2$		Result1

Table 9

As another example, Table 10 shows the separate multiplication and accumulation of sets A and B and sets C and D, where each of these sets includes 4 data elements.

Multiply-Add Source1, Source2				
A ₁	A ₂	C ₁	C ₂	Source1
B ₁	B ₂	D ₁	D ₂	Source2
=				
A ₁ B ₁ +A ₂ B ₂		C ₁ D ₁ +C ₂ D ₂		Result1

Multiply-Add Source3, Source4				
A ₃	A ₄	C ₃	C ₄	Source3
B ₃	B ₄	D ₃	D ₄	Source4
=				
A ₃ B ₃ +A ₄ B ₄		C ₃ D ₃ +C ₄ D ₄		Result2

Packed Add Result1, Result2		
A ₁ B ₁ +A ₂ B ₂	C ₁ D ₁ +C ₂ D ₂	Result1
A ₃ B ₃ +A ₄ B ₄	C ₃ D ₃ +C ₄ D ₄	Result2
=		
A ₁ B ₁ +A ₂ B ₂ +A ₃ B ₃ +A ₄ B ₄	C ₁ D ₁ +C ₂ D ₂ +C ₃ D ₃ +C ₄ D ₄	Result6

Table 10

3) Dot Product Algorithms

Dot product (also termed as inner product) is used in signal processing and matrix operations. For example, dot product is used when computing the product of matrices, digital filtering operations (such as FIR and IIR filtering), and computing correlation sequences. Since many speech compression algorithms (e.g., GSM, G.728, CELP, and VSELP) and Hi-Fi compression algorithms (e.g., MPEG and subband coding) make extensive use of digital filtering and correlation

computations, increasing the performance of dot product increases the performance of these algorithms.

The dot product of two length N sequences A and B is defined as:

$$\text{Result} = \sum_{i=0}^{N-1} A_i \cdot B_i$$

5

Performing a dot product calculation makes extensive use of the multiply accumulate operation where corresponding elements of each of the sequences are multiplied together, and the results are accumulated to form the dot product result.

The dot product calculation can be performed using the multiply-add instruction. For example if the packed data type containing four sixteen-bit elements is used, the dot product calculation may be performed on two sequences each containing four values by:

- 1) accessing the four sixteen-bit values from the A sequence to generate Source1 using a move instruction;
- 15 2) accessing four sixteen-bit values from the B sequence to generate Source2 using a move instruction; and
- 3) performing multiplying and accumulating as previously described using a multiply-add, packed add, and shift instructions.

For vectors with more than just a few elements the method shown in Table 10 is used and the final results are added together at the end. Other supporting instructions include the packed OR and XOR instructions for initializing the accumulator register, the packed shift instruction for shifting off unwanted values at

the final stage of computation. Loop control operations are accomplished using instructions already existing in the instruction set of processor 109.

4) Discrete Cosign Transform Algorithms

Discrete Cosine Transform (DCT) is a well known function used in many signal processing algorithms. Video and image compression algorithms, in particular, make extensive use of this transform.

In image and video compression algorithms, DCT is used to transform a block of pixels from the spatial representation to the frequency representation. In the frequency representation, the picture information is divided into frequency components, some of which are more important than others. The compression algorithm selectively quantizes or discards the frequency components that do not adversely affect the reconstructed picture contents. In this manner, compression is achieved.

There are many implementations of the DCT, the most popular being some kind of fast transform method modeled based on the Fast Fourier Transform (FFT) computation flow. In the fast transform, an order N transform is broken down to a combination of order N/2 transforms and the result recombined. This decomposition can be carried out until the smallest order 2 transform is reached. This elementary 2 transform kernel is often referred to as the butterfly operation. The butterfly operation is expressed as follows:

$$X = a*x + b*y$$

$$Y = c*x - d*y$$

where a, b, c and d are termed the coefficients, x and y are the input data, and X and Y are the transform output.

The multiply-add allows the DCT calculation to be performed using packed data in the following manner:

- 5 1) accessing the two 16-bit values representing x and y to generate Source1 (see Table 11 below) using the move and unpack instructions;
- 2) generating Source2 as shown in Table 11 below -- Note that Source2 may be reused over a number of butterfly operations; and
- 3) performing a multiply-add instruction using Source1 and Source2 to generate
10 the Result (see Table 11 below).

x	y	x	y	Source1
a	b	c	-d	Source2
$a \cdot x + b \cdot y$		$c \cdot x - d \cdot y$		Source3

Table 11

In some situations, the coefficients of the butterfly operation are 1. For these cases,
15 the butterfly operation degenerates into just adds and subtracts that may be performed using the packed add and packed subtract instructions.

An IEEE document specifies the accuracy with which inverse DCT should be performed for video conferencing. (See, IEEE Circuits and Systems Society, "IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine
20 Transform," IEEE Std. 1180-1990, IEEE Inc. 345 East 47th St., NY, NY 10017,

USA, March 18, 1991). The required accuracy is met by the disclosed multiply-add instruction because it uses 16-bit inputs to generate 32-bit outputs.

In this manner, the described multiply-add instruction can be used to improve the performance of a number of different algorithms, including algorithms that
5 require the multiplication of complex numbers, algorithms that require transforms, and algorithms that require multiply accumulate operations. As a result, this multiply-add instruction can be used in a general purpose processor to improve the performance of a greater number algorithms than the described prior art instructions.

ALTERNATIVE EMBODIMENTS

10 While the described embodiment uses 16-bit data elements to generate 32-bit data elements, alternative embodiments could use different sized inputs to generate different sized outputs. In addition, while in the described embodiment Source 1 and Source 2 each contain 4 data elements and the multiply-add instruction performs two multiply-add operations, alternative embodiment could operate on packed data
15 having more or less data elements. For example, one alternative embodiment operates on packed data having 8 data elements using 4 multiply-adds generating a resulting packed data having 4 data elements. While in the described embodiment each multiply-add operation operates on 4 data elements by performing 2 multiplies and 1 addition, alternative embodiments could be implemented to operate on more
20 or less data elements using more or less multiplies and additions. As an example, one alternative embodiment operates on 8 data elements using 4 multiplies (one for each pair of data elements) and 3 additions (2 additions to add the results of the 4 multiplies and 1 addition to add the results of the 2 previous additions).

While the invention has been described in terms of several embodiments, those skilled in the art will recognize that the invention is not limited to the embodiments described. The method and apparatus of the invention can be practiced with modification and alteration within the spirit and scope of the appended claims. The
5 description is thus to be regarded as illustrative instead of limiting on the invention.

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220